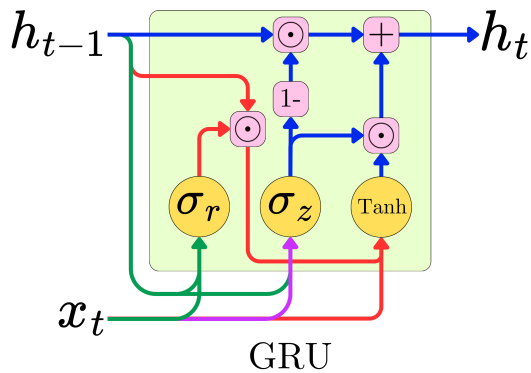
Understanding the Gated Recurrent Units (GRU)

Damien Benveniste



GRU

The Gated Recurrent Unit (GRU) was introduced in 2014 by Cho et al. as a simpler alternative to the LSTM unit. As the LSTM, it is designed to address the vanishing gradient problem and improve sequence modeling while simplifying the structure of LSTM networks by using fewer gates and maintaining similar performance.

1 The Architecture

The GRU cell uses as input the current input $\mathbf{x}_t$ from the data at the time-step t, and the hidden state of the previous iteration $\mathbf{h}_{t-1}$. The short and long-term dependencies are captured both by the hidden state $\mathbf{h}$. We have only two mixing gates:

- The update gate:

$$\mathbf{z}_t = \sigma\left(W_z\mathbf{x}_t + U_z\mathbf{h}_{t-1} + \mathbf{b}_z\right) \tag{1}$$

- and the reset gate:

$$\mathbf{r}_t = \sigma\left(W_r\mathbf{x}_t + U_r\mathbf{h}_{t-1} + \mathbf{b}_r\right) \tag{2}$$

Again, $\sigma(\cdot)$ is the logistic function.

By using the input $\mathbf{x}_t$, the previous hidden state $\mathbf{h}_{t-1}$ and the reset gate $\mathbf{r}_t$, we are generating a candidate hidden state:

$$\tilde{\mathbf{h}}_t = \tanh\left(W_h\mathbf{x}_t + U_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h\right) \tag{3}$$

The reset gate $\mathbf{r}_t$ decides whether to ignore parts of $\mathbf{h}_{t-1}$. If $\mathbf{r}_t \approx 0$, the model discards irrelevant past information, and the hidden state is reset, and if $\mathbf{r}_t \approx 1$, the cell remembers the full past information.

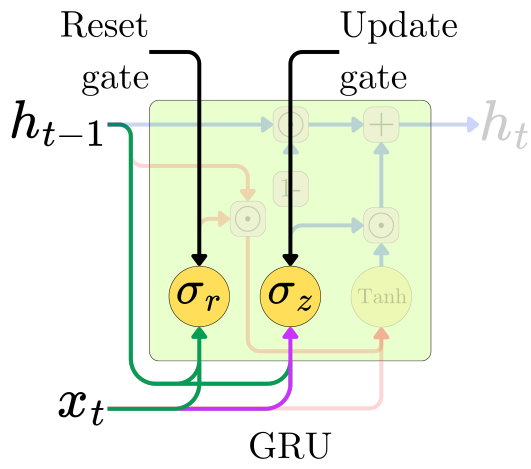


Figure 1: The inputs $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ are mixed through two gates: the update gate and the reset gate.

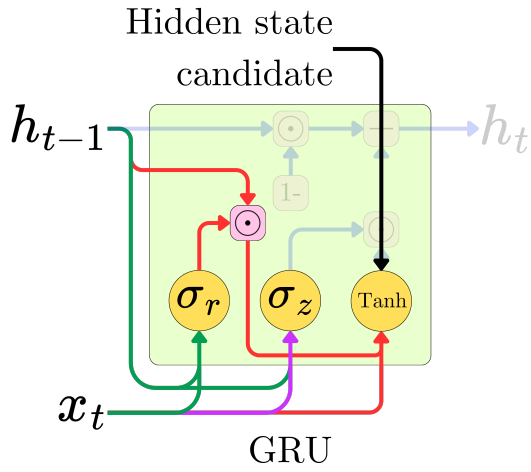


Figure 2: The hidden state candidate will allow the cell to inject new information from the current input.

The final hidden state is chosen as a weighted average as the new hidden state candidate $\tilde{\mathbf{h}}_t$ and the previous hidden state $\mathbf{h}_{t-1}$:

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \tag{4}$$

$\mathbf{z}_t$ acts like a "slider" that adjusts between short-term memory (current input) and long-term memory (past hidden states). If $\mathbf{z}_t \approx 1$, the GRU replaces $\mathbf{h}_{t-1}$ with $\tilde{\mathbf{h}}_t$ and if $\mathbf{z}_t \approx 0$, it retains the previous state $\mathbf{h}_{t-1}$.

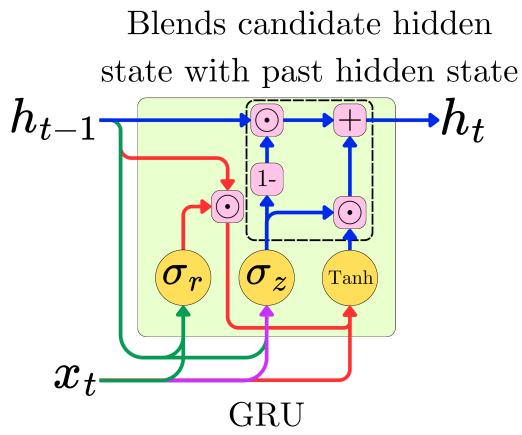


Figure 3: The update gate allows to balance the old information with the new one.

2 Long-Term Memory VS Short-Term Memory

Let's draw a simple argument on why this architecture prevents vanishing and exploding gradients. In the extreme case $\mathbf{z}_t \approx 0$ where the model learns to preserve the long-term memory, we have the asymptotic behavior of gradients in one dimension:

$$\frac{\partial z_t}{\partial h_{t-1}} \approx 0 \tag{5}$$

This is due to the specific analytical form of the logistic function in that region.

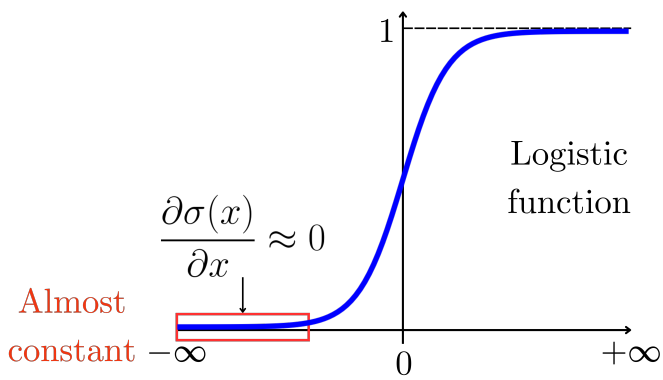


Figure 4: The logistic function is almost constant in the asymptotic regions.

This means that:

$$\frac{\partial h_t}{\partial h_{t-1}} \approx 1 - z_t \tag{6}$$

When we unroll this RNN for T time steps, the hidden states again form a dependency chain:

$$\mathbf{h}_1 \to \mathbf{h}_2 \to \cdots \to \mathbf{h}_T \tag{7}$$

and we have

$$\frac{\partial h_T}{\partial h_1} \approx \prod_{t=2}^{T} (1 - z_t) \tag{8}$$

with $1 - z_t \lesssim 1$ for every iteration, the product decays slowly with the number of iterations and cannot explode. This provides the mechanism to capture the memory for long-term dependencies. In the other extreme, we have $\mathbf{z}_t \approx 1$ and

$$\frac{\partial h_t}{\partial h_{t-1}} \approx z_t \frac{\partial \tilde{h}_t}{\partial h_{t-1}} \tag{9}$$

The partial derivative $\frac{\partial \tilde{h}_t}{\partial h_{t-1}}$ can be further decomposed in its contribution from $\frac{\partial r_t}{\partial h_{t-1}}$. As in the case of LSTM, the gradient route from $\mathbf{h}_T$ to $\mathbf{h}_1$ accumulates multiple gating factors bounded within $[0, 1]$. That is why, over many steps, the signal is likely to decay pretty fast instead of blowing up, and $\mathbf{z}_t \approx 1$ sets the network up for capturing short-term dependencies.